Lighting Functions for Realistic Images

Richard A. Redner, Mark E. Lee and Samuel P. Uselton

RNR Technical Report RNR-91-005, January 1991

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

ARC 275 (Rev Feb 81)

Lighting Functions for Realistic Images

Richard A. Redner, Mark E. Lee and Samuel P. Uselton

RNR Technical Report RNR-91-005, January 1991

Richard Redner is with the University of Tulsa in Tulsa Oklahoma and Mark Lee is employed by Amoco Production Company, Tulsa Research Center. Sam Uselton is employed by Computer Sciences Corporation. His work was supported under NASA contract NAS 2-12961.

This work was also partially supported by the National Science Foundation under grant number CCR-8915693.

Lighting Functions for Realistic Images

Richard A. Redner, Mark E. Lee and Samuel P. Uselton

January, 1991

Abstract

We present a new method for representing light sources for use in realistic image synthesis. Each light source is described as a multi–dimensional function based on product tensor splines. This representation allows the radiance of the light sources to vary according to the direction of emission, wavelength, and position on the emitting surface. The product tensor spline lighting functions are easily evaluated, can be generated from sample rays and can generate samples for distributing light throughout the scene.

The product tensor spline lighting functions have been incorporated into a wavelength dependent bidirectional ray tracing program and example images are provided which exhibit prismatic effects.

Keywords: bidirectional ray tracing, lighting functions, sampling.

1. Introduction

Many recently developed algorithms for the generation of realistic computer generated images can naturally be decomposed into two procedures. The first procedure maps the initial lighting information into the scene and the second procedure performs the final rendering. As an intermediary between these two procedures we have a function or structure which captures this complex lighting information.

Complex lighting functions can also be used as primary sources. In particular the addition of distributed light sources into ray tracing systems has greatly improved the quality of ray traced images.

In this paper we will briefly review some of the papers which have considered distributed and directionally dependent light sources. We will then review several 2-pass procedures related to ray tracing and radiosity and comment on the lighting function or structure used to connect the light propagation system with the rendering system. We will describe our wavelength dependent bidirectional ray tracing system and develop a new algorithm for the representation of lighting information.

These ideas have been used to construct a wavelength dependent bidirectional ray tracing system and we end the paper by describing 2 images which exhibit prismatic effects.

2. Related work

The need for more complex primary light sources has been addressed by several authors. In particular, Warn [Warn83] developed easily implemented directionally dependent lights and implemented flaps and cones in order to create spotlights. The flaps and cones are used to restrict the influence of the source when shadow rays are sent to the light. Verbeck [Verb84] and Nishita et. al. [Nish85] have used point light sources which vary in

intensity and spectral content as a function of direction. Complex lighting functions simulating realistic outdoor lighting conditions were used by Nishita and Nakamae [Nish86] and Takagi et. al. [Taka90].

Cook et. al [Cook84] introduced distributed lights or area lights to ray tracing and light sources used in radiosity are usually area light sources. As was pointed out by Kajiya [Kaji86], both ray tracing and radiosity methods represent methods for solving a particular integral equation commonly referred to as the rendering equation. For more information on algorithms used to solve the rendering equation and an excellent presentation on light transport and the necessities of hybrid solutions, please refer to the papers by Sillion and Puech [Sill89], Heckbert [Heck90] and Watt [Watt90].

The radiosity method was introduced by Goral et. al. [Gora84] and is a direct attempt to solve a discretized version of the radiosity equation. Surfaces are partitioned into grids and each cell is assigned a radiosity value by the light propagation algorithm. The radiosity algorithm was improved by Cohen and Greenberg in [Cohe85] by the introduction of the hemi-cube structure. The radiosity paradigm was further extended to include specular reflection [Imme86] and the hemi-cube structure was replaced by global cubes to allow for the addition of directional information. In the paper by Wallace et. al. [Wall87] a "two-pass solution" is described where the light propagation algorithm computes the diffuse lighting effects and the specular lighting effects are determined by a ray tracing procedure. This procedure was extended by Sillion and Puech [Sill89]. Again the intermediate lighting function is associated with a discrete grid structure. Finally the paper by Campbell and Fussell [Camp90] introduces adaptive mesh generation in order to improve the quality and efficiency of radiosity generated images. All of these methods are based on some sort of mesh generation to store and represent the intermediate lighting function.

A different approach to the light propagation problem was introduce by Watt [Watt90]. Watt introduced the use of backward beam tracing, i.e. beam tracing from the light, and this was used to capture caustics and several new and interesting lighting effects. The lighting information from the beam tracing is represented as polygonal illumination volumes and thus has the advantage in that the structure can capture rapid changes in illumination without imposing a uniformly small grid. The final image is rendered using a ray tracing system.

Ray tracing is a very powerful method for generating images which exhibit specular effects but one of its weaknesses is the expense involved in capturing the effects of diffuse illumination. Arvo [Arvo86] suggested that ray tracing could be performed in the forward and backward direction in order to capture diffuse lighting effects. Heckbert [Heck90] described a bidirectional ray tracing system based on adaptive radiosity textures. Again we see the use of a simple but adaptive mesh to store the intermediate lighting information. However, in the same paper, he put forward the idea that a lighting function is a density and that histograms or kernel estimates could be used to estimate density functions. This opens the door for the use of other types of nonparametric density estimators.

The difficulty with using histograms is that they are discontinuous functions and must be filtered or interpolated for use in computer graphics. The radiosity solutions are also histograms that are in fact interpolated for use in rendering. These estimators can be

thought of as first order splines.

The problem with kernel density estimates is that, if we let N denote the number of samples, then storage is $O(N)$ and evaluation at many points is best done by performing a sort which is $O(N \ln N)$. Evaluation is then $O(\ln N)$. If you don't sort, evaluation is $O(N)$. Our limited experiments with kernel density estimators revealed that the creation of quality images required a prohibitively large number of kernels. This makes the estimator impractical for graphics applications. Difficulties with edge effects were also experienced. Unfortunately, at the time of Heckbert's paper, no suitable alternative nonparametric density estimation scheme was available.

In this paper we describe a non–parametric density estimator based on product tensor splines in which storage is $o(N)$ and evaluation is performed in constant time with the use of uniform structures. Random values from the density estimate can be generated in a very efficient manner.

3. A BI-DIRECTIONAL RAY TRACING PROGRAM

Let S be a surface in R^3 and I be an interval of real numbers corresponding to the wavelengths of visible light. Let H be the hemisphere of directions

$$H = \{(x, y, z) | z \geq 0 \quad x^2 + y^2 + z^2 = 1\}.$$

We define a lighting function as a function

$$l : S \times I \times H \rightarrow R^+$$

the non-negative real numbers.

The function l represents the lighting function specified by the particular graphics application . The function can represent intensity for earlier rendering systems based on local illumination models [Blin77,Cook82]. For our application we think of a lighting function as representing the energy per unit time per unit area per unit interval of wavelength per unit solid angle and thus l is an energy density function. The total energy per unit time is defined by

$$E = \int_S \int_I \int_H l \, d\omega \, d\lambda \, dA$$

and thus l/E is a density function in the sense of probability theory.

Before we continue further, we observe that lighting functions may be defined over other domains. In particular S may be a volume, a curve or a point. The color domain may be an interval larger than the interval of visible light or may be one of the usual 3 dimensional color spaces. As S changes, the set of admissible directions H may also be suitably modified. All of our comments in this paper are fairly easily extended to the multiplicity of lighting domains.

Given a scene to render, we identify the lighting functions in that scene with density functions and generate random rays whose statistical distribution is that of the density function. Each ray carries with it a wavelength value and an energy value. The wavelength

is determined randomly according to the energy distribution of the light source. In order that most of the rays intersect the object or objects which have been targeted, a bounding volume is created about the important objects and a "cone" is created which contains the light source at one end and the bounding volume at the other.

At this time we have implemented point sources as the primary sources of light and have used spherical bounding volumes. We then generate light rays which lie in this light cone. Every ray which is generated is guaranteed, by its construction, to lie in this cone. To increase the sampling efficiency, stratification of the direction of light rays is used. The details of this computation are presented in Appendix 1. Stratification in wavelength sampling is also used to insure color balance and to improve efficiency.

The energy associated with each ray depends on the number of rays, N, the aperture of the cone which is determined by an angle, ϕ_0, and the total energy, E, of the source. The energy associated with the ray is therefore given by

$$E_{ray} = \frac{E}{N} \frac{2\pi(1 - \cos(\phi_0))}{4\pi}$$

where the second factor is the fraction of the area of the sphere of radius one which lies inside the cone.

In order for this to be consistent with the ray tracing portion of the program, the lighting computation in the ray tracer must account for the distance from the point on the surface to the light source. In particular, if distance is denoted by r, the factor of

$$\frac{1}{4\pi r^2}$$

must be included in the traditional ray tracing computation when a ray is shot towards a light. This is a well known fact [Kaji90] but it is frequently ignored.

In the light propagation portion of the algorithm, two special types of objects are recognized. The algorithm requires a list of objects at which light rays are targeted and a list of objects upon which lighting functions will be built. For each target object and for each light source, rays are generated towards the targeted object.

Those rays which strike the targeted objects are allowed to propagate through the scene. If they hit any of the objects upon which a lighting function is to be built, then the lighting function for that objects is updated. Upon completion of the light propagation algorithm, the lighting functions are written to a file. A traditional ray tracing program is applied to the original scene with the lighting functions added as internal lighting sources. Care is taken so that no redundancy occurs in the accumulation of lighting information.

Depending on the specifics of the implementation, prismatic effects, the focusing of lenses, color bleeding and indirect lighting can be demonstrated.

4. Density estimation

We now describe a convenient structure for the representation of lighting functions over surfaces. We will describe how this lighting function can be determined from data and also how it can be used in a distributed ray tracing system.

A convenient representation for lighting functions can be constructed out of tensor product splines but for the sake of this exposition we will first consider the standard density estimation problem and then we will return to the lighting function estimation problem. The ideas and theorems presented in this section have been developed for the explicit purpose of estimating lighting functions in computer graphics applications. Let S denote a subset of some rectangle R in the $x-y$ plane and suppose that we wish to generate a density function over S given data. Given a sequence of knots in both the x and y directions and a natural number $m \geq 1$ we define $l(x,y)$ to be a product tensor spline of order m and hence

$$l(x,y) = \sum_i \sum_j c_{ij} B_i(x) B_j(y).$$

The functions $B_i(x)$ and $B_j(y)$, for a suitable range of values of i and j, are B–spline basis functions and are completely specified by the knot sequence. The B–spline basis functions are non-negative at every point and the sum of all the basis functions at a point is always one [Bart87]. These two properties make B–spline basis functions particularly useful for the estimation of densities, and since we will be working with density functions it is convenient to introduce normalizing constants

$$h_{ij} = \int_S \int B_i(x) B_j(y)\, dx\, dy$$

and write

$$l(x,y) = \sum_i \sum_j \alpha_{ij} \frac{B_i(x) B_j(y)}{h_{ij}}.$$

Since the B–splines basis functions are non-negative, $l(x,y)$ is a density function if all the α_{ij} are non-negative and if $\sum_i \sum_j \alpha_{ij} = 1$.

We now construct the density estimate given data from the density f on S. Let $\{(x_k, y_k)\}_{k=1}^N$ be identically distributed random variables from some fixed but unknown density function f on S. Choose a sequence of knots in the x and y directions whose spacing depends on N and let $B_i(x)$ and $B_j(y)$ denote the corresponding B–splines basis functions. Let $\hat{h}(N) = \max_{ij} h_{ij}$. and $h(N) = \min_{ij} h_{ij}$. We assume that the knot spacing is chosen so that (1) the knot spacing in the x and y directions goes to zero as N goes to infinity, (2) there is a number C independent of N so that $\hat{h}(N)/h(N) \leq C$ and (3) that $Nh(N) \to \infty$ as $N \to \infty$.

Define for i and j,

$$\alpha_{ij} = 1/N \sum_i \sum_j B_i(x_k) B_j(y_k)$$

and

$$l_N(x,y) = \sum_i \sum_j \alpha_{ij} \frac{B_i(x) B_j(y)}{h_{ij}}.$$

Theorem 1. *If (x,y) is in S and if f is a bounded continuous function over S, then $E(l_N(x,y) - f(x,y))^2 \to 0$ as $N \to \infty$ in which case we say that $l_N(x,y)$ converges in mean square error (MSE) to $f(x,y)$.*

This result is easily extended to provide results for parametrically defined surfaces. In particular let $\mathbf{r} : S \to R^3$ be a smooth 1-1 function. Then $M \equiv \mathbf{r}(S) = \{X = \mathbf{r}(u,v) | (u,v) \in S\}$ is a parameterized surface. Since $\mathbf{r}$ is 1-1, we define the function $(u(X), v(X)) : M \to S$ as the inverse function of $\mathbf{r}$ and define a lighting function of M in terms of a lighting function on S. Given a sample of random points $\{X_k\}_{k=1}^N$ from a density function f on M, we define a lighting function on S by

$$\alpha_{ij} = 1/N \sum_i \sum_j B_i(u(X_k)) B_j(v(X_k))$$

and

$$l_N(x,y) = \sum_i \sum_j \alpha_{ij} \frac{B_i(x) B_j(y)}{h_{ij}}.$$

We then define a density function on M as

$$l_N^M(X) = l_N(u(X), v(X)) / || \frac{\partial r}{\partial u} \times \frac{\partial r}{\partial v} ||$$

Theorem 2. *Suppose that the conditions on the knots and basis functions from theorem 1 are satisfied and suppose that there are numbers a and b so that*

$$||\partial \mathbf{r}/\partial u|| \leq b$$

$$||\partial \mathbf{r}/\partial v|| \leq b$$

and

$$0 < a \leq ||\partial \mathbf{r}/\partial u \times \partial \mathbf{r}/\partial v|| \leq b$$

If f is continuous and bounded over M then l_N^M converges in mean square error to the density function f on M.

The proof of these results can be found in two papers by Redner and Gehringer [Gehr91,Redn91].

5. Density estimation on the sphere

These results are applicable as long as the manifold M has the same topology as some subset S of the plane. However, for our lighting function we wish to represent directions as points on a sphere. Since a sphere is not topologically equivalent to any subset of the plane we must construct a new density estimator for spheres. We can, however, build upon the previous results to obtain a density estimator for the sphere. We begin by considering the simpler problem of estimating a density function on the unit circle $C = \{(x,y) | x^2 + y^2 = 1\}$. Given $m > 1$ and a uniform knot sequence, we can form the uniform B–spline density estimator on the interval $[0, 2\pi]$. If the ends of the interval are identified (i.e. the points are paired up) we obtain a density estimator on the circle, but of course this estimate is discontinuous. We fix this by identifying (in order of increasing indices) the first and

last $m-1$ basis functions. Now when the interval $[0, 2\pi]$ is mapped to the circle, there is no ambiguity at 0 and 2π and furthermore the function exhibits the same degree of smoothness at the image of 0 and 2π as is does at all of the other knots.

From an operational point of view the density estimate can be obtained in the following easy fashion. Given data on the unit circle we use the following procedure

1. find the preimage of the data (which are points in $I = [0, 2\pi]$).
2. compute the B–spline coefficients for a density estimator on the interval I.
3. since the the first and last $m-1$ B–spline coefficients are identified, add to the first $m-1$ coefficients the coefficient of the corresponding B–spline coefficients at the right end.
4. replace the last $m-1$ coefficients with the corresponding coefficients from the first $m-1$ B–spline coefficients.
5. map this density estimator back to the circle.

Observe that the weights (the h's) will be the same for each B–spline, since we took the sum of the coefficients for the corresponding B–spline. The resulting density estimator will be $m-2$ times differentiable on the circle.

This can be extended to the cylinder in the obvious manner. Of course we require that the knots lie on a grid and that the spacing be uniform in the x direction. Arbitrary knot spacing may be used in the y direction since no identification of splines is required along the top and bottom of the cylinder.

To extend this to the hemisphere we make one additional modification. We observe that the top line of the rectangle is mapped to the north pole. In order that the density be continuous at the north pole we identify rows of B–splines along the top of the rectangle. More specifically, given $m > 1$ we identify all of the B–splines in each of the top $m-1$ rows of the rectangle.

From an operation standpoint, we can use the 2-D density estimation routines to create and evaluate densities on the hemisphere. Given $m > 1$ and a 2 dimensional grid of knots which have uniform spacing in x and given a set of data on the hemisphere we use the following procedure

1. find the preimage of the data (which are points in $R = [0, 2\pi] \times [0, 1]$).
2. compute the B–spline coefficients for a density estimator on the rectangle R.
3. since the the first and last $m-1$ B–spline coefficients are identified for each row, add to the first $m-1$ coefficients the coefficient of the corresponding B–spline coefficients in each row.
4. replace the last $m-1$ coefficients in each row with the corresponding coefficients from the first $m-1$ B–spline coefficients in that row.
5. for each of the top $m-1$ rows replace the B–spline coefficients with the average of the coefficients for the B–splines in that row. Do not use that last $m-1$ coefficients since that would be double counting.
6. map this density estimator back to the hemisphere using the function $D(\theta, t) = (\sin(\theta)\sqrt{1-t^2}, \cos(\theta)\sqrt{1-t^2}, t)$

The resulting density estimate on the hemisphere will be $m-2$ times differentiable except possibly at the poles where the density is continuous.

Extension to the entire sphere is accomplished in a similar fashion. We use a larger rectangle, $[0, 2\pi] \times [-1, 1]$, and use the hemisphere modification to the cylinder procedure at both the top and the bottom of the rectangle.

In order for the density estimation procedure to as efficient as possible it is desirable for the supports of the basis functions have approximately the same area and that they be as 'round' as possible. In the following table, h is the height of the top section of the rectangle and a and b are the width and height of the remaining rectangles. The numbers have been chosen so that each cell has approximately the same area and so that $a \approx b$. The following values are obtained from the formulas $h = 1/(ij^2 + 1)$, $b = ij/h$, and $a = 2\pi/ij$. The number of cells will be $ij^2 + 1$.

i	j	h	b	a	number of cells
5	1	1/6	5/6	$2\pi/5$	6
6	1	1/7	6/7	$\pi/3$	7
7	1	1/8	7/8	$2\pi/7$	8
5	2	1/21	10/21	$2\pi/10$	21
6	2	1/25	12/25	$\pi/6$	25
7	2	1/29	14/29	$2\pi/14$	29
5	3	1/46	15/46	$2\pi/15$	46
6	3	1/55	18/55	$\pi/9$	55
7	3	1/64	21/64	$2\pi 21$	64

For spline of degree m, the knots in the plane which correspond to these regions are

$$x \in \{-(m-1)a, \ldots, -a, 0, a, \ldots, 2\pi, 2\pi + a, \ldots, 2\pi + (m-1)a\}$$

$$y \in \{-(m-1)b, \ldots, -b, 0, b, \ldots, 1 - h, 1, 1 + h, \ldots, 1 + (m-1)h\}.$$

So for example, if $m = 3$, $i = 5$ and $j = 1$ then the knots satisfy

$$x \in \{-2a, -a, 0, a, 2a, 3a, 4a, 5a = 2\pi, 6a = 2\pi + a, 7a = 2\pi + 2a\}$$

$$y \in \{-2b, -b, 0, b = 1 - h, 1, 1 + h, 1 + 2h\}.$$

6. Estimation of lighting functions

It is an easy step from the estimation of density functions to the the estimation of lighting functions. Given a set of rays, randomly generated from a light source, these rays strike the surface of interest. Suppose that the rays strike a diffuse planar surface S and suppose that we are working in wavelength. A lighting function can be described in terms of a three-dimensional tensor spline. Each ray emanating from the light source, carries with it an energy value and a wavelength value. Let (x_k, y_k), $k = 1, \ldots, N$ denote the coordinates at which the rays strike the surface and let λ_k and E_k, $k = 1, \ldots, N$ denote the wavelength and the energy associated with the kth ray. Set up a system of knots in

the x, y and λ directions which define basis functions $B_i(x)$, $B_j(y)$ and $B_l(\lambda)$. We then define

$$\alpha_{ijl} = 1/N \sum_i \sum_j \sum_l E_k B_i(x_k) B_j(y_k) B_l(\lambda_k)$$

$$h_{ijl} = \int_S \int \int_I B_i(x) B_j(y) B_l(\lambda)\, d\lambda\, dA$$

and

$$l_N(x, y, \lambda) = \sum_i \sum_j \sum_l \alpha_{ijl} \frac{B_i(x) B_j(y) B_l(\lambda)}{h_{ijl}}.$$

Use of these tensor splines as lighting functions is extremely efficient in terms of both storage and computational speed. Procedurally, the unknown coefficients (the α's) are initially set to zero and the scaling coefficients (the h_{ijl}'s) are determined. In the rectangular case each h_{ijl} is the product of three factors from integrals in the x, y and λ directions. These values are easily computed since we are working with splines. When S is not rectangular each h_{ijl} factors into an integral over S times an integral over wavelength.

The computational expense of using tensor spline lighting functions is minimal. The lighting functions are as smooth as the user specifies and storage is proportional to the total number of basis functions. Once initialized, the tensor spline lighting function can be updated in constant time for uniform knot spacing and (due to the binary searches which are used) logarithmic time in the number of knots for arbitrary knot spacing.

The extension of this procedure to directionally dependent lighting functions is carried out by extending the function to five dimensions using the product of three dimensional tensor product splines for space and wavelength and the modified two dimensional splines developed for the hemisphere. The extension to directionally dependent lighting functions over smooth surfaces is straightfoward using the material from section 4. We recall that many lighting functions vary slowly over the surface when you use distributed light sources and most of the sudden changes in brightness occur at boundaries of objects. Therefore the number of knots used in each coordinate direction should be reasonable.

7. Tensor lighting functions as distributed light sources

In a distributed ray tracing system, area light sources are sampled to illuminate the scene. It is required then, that we be able to generate random points from the tensor product lighting functions. In the case that we have used a uniform knot spacing in each coordinate direction, this is a very manageable problem.

We begin by observing that if $x_1, x_2, \ldots x_m$ are independent and uniformly distributed over the interval $[0, 1]$ then the density function for the random variable $x = x_1 + x_2 + \cdots + x_m$ is a uniform B–spline of order m. Our diffuse tensor product lighting function over a rectangle has the form

$$l_N(x, y, \lambda) = \sum_i \sum_j \sum_l \alpha_{ijl} \frac{B_i(x) B_j(y) B_l(\lambda)}{h_{ijl}}$$

where

$$h_{ijl} = \int_S \int \int_I B_i(x) B_j(y) B_l(\lambda)\, d\lambda\, dA$$

and

$$\alpha_{ijl} = 1/N \sum_i \sum_j \sum_l E_k B_i(x_k) B_j(y_k) B_l(\lambda_k).$$

If a uniform knot spacing is used in each of the coordinate directions, then each basis function is a scaled translated version of the standard basis function on the interval $[0, m]$. To generate a random point from such a light source, select indices ijl with probability α_{ijl} (this is done using a summed area table) and generate a random point (x, y, λ) by selecting random points from the densities $B_i(x)$, $B_j(y)$ and $B_l(\lambda)$. If a directionally dependent lighting function has been created and again if the knots are uniformly spaced in the direction of each of the coordinate axis, we generate random points on the hemisphere using the same procedure and the mapping $D(\theta, t) : [0, 2\pi] \times [0, 1] \rightarrow H$. For parametrically defined surfaces, random points are generated in the plane and mapped to the parametrically defined surface.

8. Examples

Image 1 is an image of the lighting function created by the light propagation algorithm. A point light source is located at a distance 600,000 units from a wall which is a unit square. The lighting energy is 3×10^{14} units. The energy distribution for the source is natural sunlight data from Wyszecki and Stiles [Wysz82]. The extreme distance was chosen so that the incoming rays are nearly parallel and the extreme value for the energy compensates for the $1/r^2$ fall off of the light energy density. The light is passed through an equilateral prism and strikes the wall.

The lighting function is made of three tensor product lighting functions (one each for of the r, g and b values) with 20 uniformly spaced knots in both coordinate directions. The image was created by applying the light propagation algorithm and simply evaluating the tensor product spline at 512×512 locations. The lighting function was zoomed by a factor of three to take this picture and still the picture is extremely smooth.

The second image results from the combined efforts of the light propagation program and a traditional ray tracing program. The scene contains the same objects as are found in image 1. Two spheres and a mirror, which is perpendicular to the first wall, have been added to the scene. An additional light source near the view point has also been added.

Notice that one of the spheres causes a shadow on the wall creating a darkened place within the spectrum. Even though the illuminating source is a point source, the spline lighting functions create a very gentle transition into the shadow and creates the illusion of a penumbra. A few "pops" are possibly noticeable due to the sampling of the spectrum as a diffuse distributed source in the ray tracing portion of the scene. These can be eliminated by a postprocessing procedure described in Lee and Redner 1990 [Lee90], by improving the sampling stratification in the ray tracing program [Lee85] or by simply increasing the sampling level. If you look closely you can see the image of the prism reflected in the dark sphere.

9. Comments and Future work

In this paper we have described tensor product lighting functions and have presented algorithms for their use and the theory which supports their application. The tensor product spline lighting functions were developed in order to provide convenient and efficient procedures for use with computer graphics systems. These lighting functions are easily evaluated, can be generated from data and can also be statistically sampled. This implies that tensor product lighting functions can be used in almost any computer graphics system and in particular can be used to replace many of the piecewise constant (first order splines) in a number of computer graphics application.

To demonstrate the effectiveness of these lighting functions we have incorporated them into a wavelength dependent bidirectional ray tracing program. Prismatic effects generated by our two pass system have been demonstrated and caustics and the focusing of lenses, color bleeding and indirect lighting affects can also be achieved using this type of system.

The areas of bidirectional ray tracing with wavelength dependent sampling and the representation of complex lighting functions are relatively new and immature areas and hence there is much work to be done. In particular, in the area of bidirectional ray tracing, there are numerous issues associated with efficiency, sampling and stratification which need to be considered. The problem of ray generation from directionally dependent sources is an interesting problem as well as the use of other types of "cone" strategies not based on the use of spherical bounding volumes. There are many possible algorithms which combine light propagation and classical ray tracing and these must be individually investigated and the possibility of iterating on the light propagation algorithms offers the promise of improved realism in computer generated images. Work is continuing on wavelength dependent models where there are interesting sampling issues involved.

The representation of complex lighting functions over surface provides numerous interesting questions. The work of Redner and Gehringer [Redn91,Gehr91] provide one set of solutions to this problem. This work is extremely general being posed in an abstract setting. Splines are not an essential part of this theory, but instead partitions of unity form the basis for the work. Splines are convenient examples of partitions of unity which are suitable for rapid and efficient computation. A number of other alternatives are possible and should be considered.

Acknowledgements

The authors would like to thank the National Science Foundation for support under grant number CCR-8915693.

Bibliography

[Arvo86] Arvo, James, "Backwards Ray Tracing", Developments in Ray Tracing, ACM SIGGRAPH 1986 Course Notes no. 12.

[Bart87] Bartels, Richard, John Beatty, and Brian Barsky, *An Introduction to Splines for use in Computer Graphics & Geometric Modeling,* Morgan Kaufmann Publishers, 1987.

[Blin77] Blinn, J.F. "Models of Light Reflection for Computer Synthesized Pictures", *Computer Graphics* 11,2 (July 1977), pp. 192-198.

[Camp90] Campbell, A.T. III, and D.S. Fussell, "Adaptive Mesh Generation for Global Diffuse Illumination", *Computer Graphics* 24,4 (Aug. 1990), pp. 155-164.

[Cohe85] Cohen, M.F., and D.P. Greenberg, "The Hemi-cube: A Radiosity Solution for Complex Environments", *Computer Graphics* 19,3 (July 1985), pp. 31-40.

[Cook82] Cook, R.L., and K.E. Torrance, "A Reflectance Model for Computer Graphics", *ACM Transactions on Graphics* 1,1 (Jan. 1982), pp. 7-24.

[Cook84] Cook, R.L., T. Porter, and L. Carpenter, "Distributed Ray Tracing", *Computer Graphics* 18,3 (July 1984), pp. 137-145.

[Gehr91] Gehringer, K.R. and R.A. Redner, "Nonparametric Density Estimation Using Tensor Product Splines", accepted subject to minor revision by *Comm. in Stat.*

[Gora84] Goral, C.M., K.E. Torrance, D.P. Greenberg and B. Battaile, "Modeling the Interaction of Light Between Diffuse Surfaces", *Computer Graphics* 18,3 (July 1984), pp. 213-222.

[Heck90] Heckbert, Paul S., "Adaptive Radiosity Textures for Bidirectional Ray Tracing", *Computer Graphics* 24,4 (Aug. 1990), pp. 145-154.

[Imme86] Immel, D.S., M.F. Cohen and D.P. Greenberg, "A Radiosity Method for Non-diffuse Environments", *Computer Graphics* 20,4 (Aug. 1986), pp. 133-142.

[Kaji86] Kajiya, James T., "The Rendering Equation", *Computer Graphics* 20,4 (Aug. 1986), pp. 143-150.

[Kaji90] Kajiya, James T., "Radiometry and Photometry for Computer Graphics", ACM SIGGRAPH Course Notes no. 24, pp. 3.1-3.30. .

[Lee85] Lee, M.E., R.A. Redner and S.P. Uselton, "Statistically Optimized Sampling for Distributed Ray Tracing", *Computer Graphics* 19,3 (July 1985), pp. 61-67.

[Lee90] Lee, M.E. and R.A. Redner, "A Note on the Use of Nonlinear Filtering in Computer Graphics", *IEEE Computer Graphics and Applications* 10,3 (May 1990), pp. 23-29.

[Nish85] Nishita, T., I. Okamura, and E. Nakamae, "Shading Models for Point and Linear Sources", *ACM Transactions on Graphics* 4,2 (April 1985), pp. 124-146.

[Nish86] Nishita and Nakamae, "Continuous Tone Representation of Three-dimensional Objects Illuminated by Skylight", *Computer Graphics* 20,4 (Aug. 1986), pp. 125-132.

[Redn91] Redner R.A., and K.R. Gehringer, "Non-parametric Density Estimation Using Partitions of Unity", to be submitted.

[Sill89] Sillion, F. and C. Puech, "A General Two-pass Method Integrating Specular and Diffuse Reflection", *Computer Graphics* 23,3 (July 1989), pp. 335-344.

[Taka90] Takagi, A., H. Takaoka, T. Oshima and Y. Ogata, "Accurate Rendering Technique Based on Colorimetric Conception", *Computer Graphics* 24,4 (Aug. 1990), pp. 263-272.

[Verb84] Verbeck, C.P. and D.P. Greenberg, "A Comprehensive Light-source Description for Computer Graphics", *IEEE Computer Graphics and Applications* 4,7 (July 1984), pp. 66-75.

[Wall87] Wallace, J.R., M.F. Cohen and D.P. Greenberg, "A Two-pass Solution to the Rendering Equation: a Synthesis of Ray-tracing and Radiosity Methods", *Computer Graphics*

21,4 (July 1987), pp. 311-320.

[Warn83] Warn, David R., "Lighting Controls for Synthetic Images", *Computer Graphics* 17,3 (July 1983), pp. 13-21.

[Watt90] Watt, M., "Light-water Interaction Using Backward Beam Tracing", *Computer Graphics* 24,4 (Aug. 1990), pp. 377-385.

[Wysz82] Wyszecki, G., and W. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae,* second edition, Wiley, New York, 1982.

APPENDIX 1.

THE GENERATION OF RAYS WITHIN A CONE

Consider the mapping

$$D : [0, 2\pi] \times [-1, 1] \rightarrow \text{ the unit sphere.}$$

defined by

$$D(\theta, t) = (\sin(\theta)\sqrt{1-t^2}, \cos(\theta)\sqrt{1-t^2}, t).$$

This function is an onto mapping which is one to one on the interior of the rectangle. Since $||D_t \times D_\theta|| = 1$ this mapping is area preserving. This implies that if (θ, t) is uniform in $[0, 2\pi] \times [-1, 1]$ then $D(\theta, t)$ is a uniform random variable on the sphere.

A word of warning is is order. It is common to write the vector

$$(\sin(\theta)\sqrt{1-t^2}, \cos(\theta)\sqrt{1-t^2}, t)$$

as

$$(\sin(\theta)\sin(\phi), \cos(\theta)\sin(\phi), \cos(\phi)).$$

Points on the unit sphere can then be generated by generating θ and $t = \cos(\phi)$ uniformly on $[0, 2\pi]$ and $[-1, 1]$ respectively, with $\sin(\phi)$ defined as $\sqrt{1-\cos^2\phi}$. If, instead, you generate ϕ uniformly on $[0, 2\pi]$, you do not get a uniform distribution on the sphere. Instead you get a distribution which is too bright at both poles.

Now we consider the problem of generating random vectors whose angle with the positive z axis is less that or equal to some angle ϕ_0. If we let $z_0 = \cos(\phi_0)$ then the region $R_{\phi_0} = [0, 2\pi] \times [z_0, 1]$ is mapped onto the intersection of the sphere with the cone pointing in the direction of the positive z-axis with angle ϕ_0.

We can therefor generate direction vectors within this cone by generating vectors in the rectangle R_{ϕ_0} and applying the mapping D.

We make the interesting observation that since the area of the rectangle $[0, 2\pi] \times [z_0, 1]$ is $2\pi(1 - z_0)$ then the area of a 'cap' of the sphere, i.e. the set of all points on the sphere with z coordinate greater than or equal to some value z_0, must also be $2\pi(1 - z_0)$. This is simply 2π times the thickness of the cap.

If the center of the cone of interest does not point in the direction of the positive z-axis, then the data generated by the above process must be rotated. If the direction of the cone

is defined by a unit vector $U = (u_1, u_2, u_3)$ then we will want a rotation which maps the vector $(0, 0, 1)$ to U. This can be done in two stages, a rotation about the y-axis followed by a rotation about the z-axis. If we let $v_3 = \sqrt{1 - u_3^2}$ then the matrices associated with these two rotations are

$$\begin{pmatrix} u_3 & 0 & v_3 \\ 0 & 1 & 0 \\ -v_3 & 0 & u_3 \end{pmatrix}$$

and

$$\frac{1}{v_3} \begin{pmatrix} u_1 & -u_2 & 0 \\ u_2 & u_1 & 0 \\ 0 & 0 & v_3 \end{pmatrix}$$

whose product (taken in the proper order of course) is

$$\frac{1}{v_3} \begin{pmatrix} u_1 u_3 & -u_2 & u_1 v_3 \\ u_2 u_3 & u_1 & u_2 v_3 \\ -v_3^2 & 0 & u_3 v_3 \end{pmatrix}$$

Stratification of samples can be used to reduce the variance of estimates of lighting function parameters. In the generation of rays from the light source, stratification can be implemented in both the direction of the rays and their associated wavelengths. The scheme outlined in section 5. can be used to stratify the ray directions. A version of this has been implemented. Stratification in wavelength is also recommended and is also implemented in our ray tracing system.

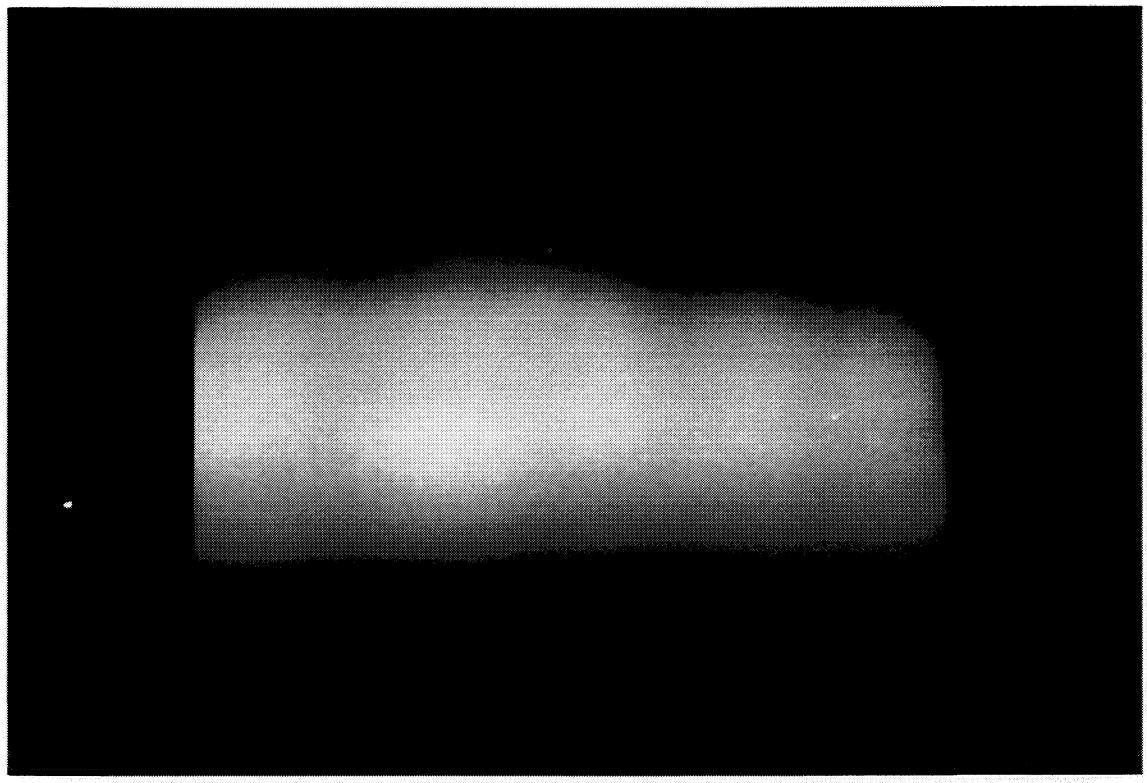

Image 1.
A spectrum created from natural sunlight

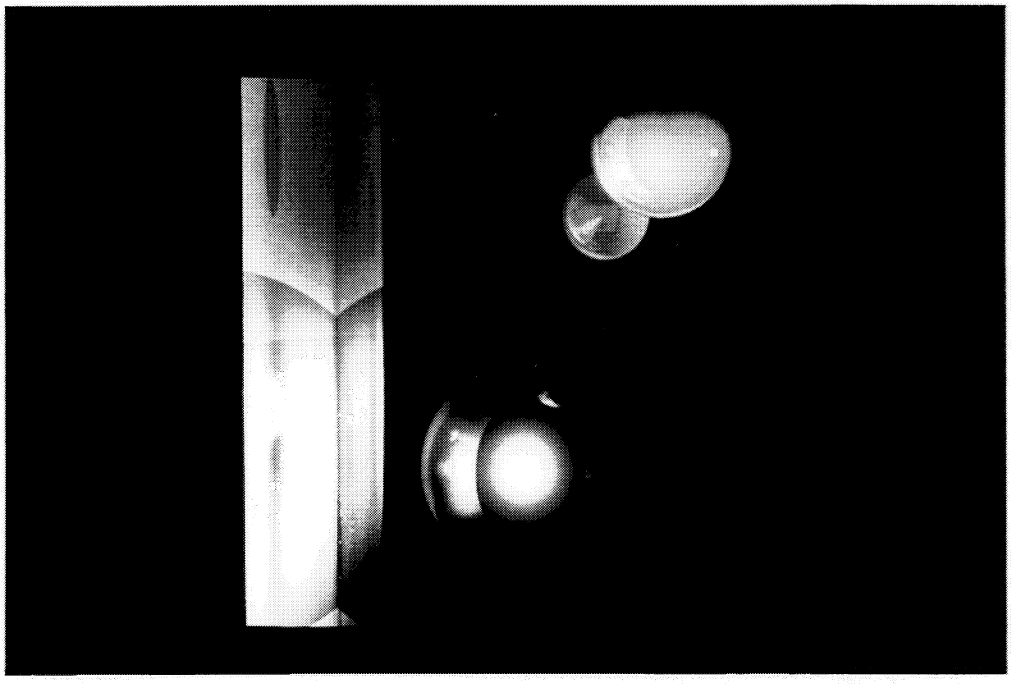

Image 2.
Spectrum Effects